Automatic Optimisation of System Architectures using EAST-ADL

Dejiu Chen, Henrik Lönn, Chokri Mraidha, Yiannis Papadopoulos, Mark-Oliver Reiser, David Servat, Luis Silva Azevedo, Sara Tucci-Piergiovanni, Martin Walker

Automatic Optimisation of System Architectures using EAST-ADL

DeJiu Chen[1], Henrik Lönn[2], Chokri Mraidha[3], Yiannis Papadopoulos[4], David Parker[4], Mark-Oliver Reiser[5], David Servat[3], Luís Silva Azevedo[4], Sara Tucci-Piergiovanni[3], Martin Walker[4]

[1] KTH Royal Institute of Technology, Brinellvägen 83, 100 44, Stockholm, Sweden
[2] Volvo Technology, Dept 6260, M2.7, 405 08 Gothenburg, Sweden
[3] CEA Saclay Nano-INNOV, Point Courrier no 174, 91191 Gif sur Yvette cedex, France
[4] University of Hull, Cottingham Road, Hull, HU67RX, UK
[5] Technische Universität Berlin, Ernst-Reuter-Platz 7, D-10587 Berlin, Germany

Abstract. With today's highly complex embedded systems, it is becoming increasingly difficult to find design solutions that meet all functional and non-functional requirements, such as performance, dependability and cost. In addition, there is often not a single optimal solution but a conflict of goals between requirements leads to a set of so-called Pareto optima. Such a multi-objective optimisation has received increasing attention in research and practice over the past few years. This paper presents current research in progress on developing a comprehensive optimisation approach that incorporates a flexible number of objectives together with the corresponding external analyses for evaluating them and uses only a single system model as information repository for all objectives and analyses. This central model is defined using the modelling language EAST-ADL, an architecture description language for the automotive domain.

1 Introduction

One of the challenging tasks in the design of complex automotive systems is balancing the many demands on the system design. The problem is amplified by the need to search a potentially huge design space, formed by the many possible configurations and variations on a design, for optimal configurations that maximise quality (i.e., dependability and performance) and minimise cost. This task can be assisted by applying model-based analysis and verification technologies to design models to answer important questions regarding the quality of individual design proposals. One practical problem here is that in complex systems, rich functionalities and their distribution across shared hardware and communication channels allow a large number of configuration options at design time and a large number of reconfiguration options at

ASCoMS, p. 1, 2013.

runtime. This creates difficulties in design because, as potential design spaces expand, their exploration for appropriate or optimal final designs suitable for implementation becomes increasingly difficult.

Whilst many design optimisation problems can only be tackled effectively by the human intellect, it is clear that, as potential design spaces expand, exploration for suitable or optimal designs (e.g. in terms of quality and cost) becomes increasingly difficult, and some automation is needed. Modelling languages and emerging architecture description languages could therefore benefit from concepts and technological support that enable this type of optimisation, while still benefiting from the support for multiple analysis & evaluation functions that an architecture description language offers. In the EU project MAENAD, techniques for such multi-objective optimisation are currently being investigated in the context of EAST-ADL — an emerging architecture description language in the automotive domain. The aim is to come up with a comprehensive, extensible approach that allows for an integration of multiple diverse optimisation objectives — and their corresponding analysis techniques and tools — into an overall automated optimisation that is based on a single, comprehensive architectural model. In addition, the optimisation approach should seamlessly integrate traditional design space variations (e.g. different bus architectures or different hardware topologies) and support for product line variability (e.g. passenger cars vs. trucks, optional equipment, country variants).

In this paper, we provide a short summary of our recent work-in-progress report on this topic [WR1] and describe the latest improvements and additions to the optimisation architecture developed in the MAENAD project.

2 Representing System Architectures with EAST-ADL

EAST-ADL is an architecture description language (ADL) intended for use in the automotive domain. It was initially defined in the European ITEA EAST-EEA project and was subsequently refined and aligned with the more recent AUTOSAR automotive standard (www.autosar.org). Responsibility for the continued development of the language now lies with the EAST-ADL Association (www.east-adl.info).

EAST-ADL enables the design and definition of automotive electronic systems by providing a comprehensive information model to capture engineering information in a standardised form. Aspects covered include vehicle features, functions, requirements, variability, software components, hardware components and communication.

One of the central features of EAST-ADL is its multi-layer approach, which defines multiple abstraction levels and distributes all development information across these levels (see Figure 1). Thus, software- and electronics-based functionality of the vehicle is described at different levels of abstraction during the development from early analysis to implementation. The intention of the abstraction levels is to provide a separation of concerns and an implicit style for using the modelling elements. The embedded system is complete on each abstraction level, and parts of the model are linked with various traceability relations. This makes it possible to trace an entity from feature level down to components in hardware and software.

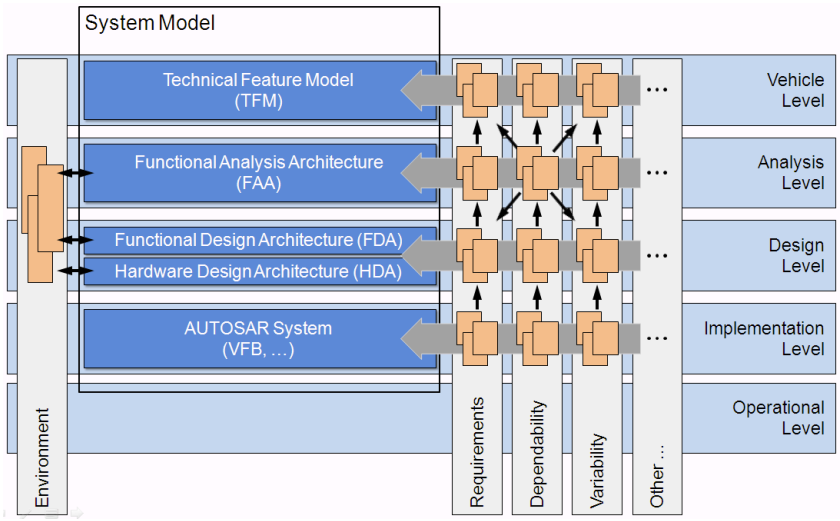

Figure 1. The EAST-ADL abstraction levels.

The top-most abstraction level is the Vehicle Level and describes the main features of the system. This is achieved primarily by the core technical feature model which represents the major functionalities and characteristics of the complete system, i.e. the entire vehicle from a top-level perspective, without exposing any realisation details. It is possible to manage the content of each individual vehicle and entire product lines in a systematic manner. In addition to this main feature model with its technical perspective, other feature models may be defined that provide views on the main feature model, e.g. a customer/marketing-oriented view defining models and packages of optional equipment.

The second level is the Analysis Level, centred on the Functional Analysis Architecture (FAA). This provides a full representation of the electronic functionality in an abstract form, the purpose of which is to define and structure of the system's functionality from a problem domain perspective, similar to a traditional functional analysis. The constituent entities of the FAA, the so-called analysis functions, capture the principal interfaces and behaviour of the vehicle's subsystems.

The third level is the Design Level, which defines the solution domain aspects across two models, the Functional Design Architecture (FDA) and the Hardware Design Architecture (HDA). The FDA defines a function architecture that takes into account hardware allocation, efficiency, legacy and reuse, commercial-off-the-shelf (COTS) availability, and other architectural qualities. The function structure is such that one or more functions can be subsequently realised by one or several AUTOSAR software components (SW-C). The external interfaces of such components correspond to the interfaces of the realised functions. On the other hand, the HDA defines the execution environment in which the software will be deployed. Its main entities are sensors, actuators, units of execution, their interfaces, and communication links be-

tween all these. The two architectures are connected by allocation relationships, which define which units of execution (as defined in the HDA) will host each software entity of the FDA.

The lowest, most concrete level is the implementation level. At this layer, EAST-ADL does not provide its own modelling entities; instead, this level is entirely defined by models from the AUTOSAR standard. In this respect, EAST-ADL can be thought of as an extension to AUTOSAR providing support for modelling on higher abstraction levels during earlier development phases. However, traceability is supported from implementation level elements (AUTOSAR) to FDA/HDA elements and, from there, further up to vehicle level elements.

Simulation or other formal techniques can be used to verify and validate a feature across all abstraction levels. This requires an environment model to be present from the early stages of the design process, and this “plant model” captures the behaviour of the vehicle dynamics, driver, etc. The core part of the environment model can be the same for all abstraction levels, which means that, for the purpose of simulation, both FAA and FDA/HDA are attached to the same environment model.

EAST-ADL also provides extensive support for managing variability, which is a key element of the EAST-ADL-based optimisation approach outlined in this paper. First, variability modelling is used to define the optimisation space (also referred to as the "architectural degrees of freedom" in [AB1]) and then configuration and variability resolution are applied to produce the optimisation candidates to be evaluated with respect to the optimisation objectives.

3 Optimisation of EAST-ADL Architectures

A great deal of information can be obtained about a system by means of model-based analysis techniques, including its dependability and timing characteristics, amongst others. This is especially true when these analysis techniques are combined with ADLs like EAST-ADL, which aim to centralise all the knowledge about a particular system. The idea is to remove the need to produce different models for different analyses, e.g. a specific error model tailored for a particular dependability analysis tool, or a separate timing model solely for timing analysis etc.

The disadvantage of this centralisation is that this wealth of information can also be difficult to manage during practical system design. A balance between the different attribute requirements can be difficult to achieve during the development of complex systems, both because there is a vast number of possible design alterations that could be made and because the interrelationships between the attributes — and therefore the effects of any alterations — are not necessarily clear. This is particularly true when tradeoffs between multiple conflicting attributes (like safety versus cost, or energy consumption versus performance) are involved, as improving one attribute may lead to a worsening of another.

Automation, e.g. of analysis techniques, helps overcome this to an extent, although this relies on the availability of compatible analysis tools. Such tools allow models to be rapidly evaluated according to different criteria, enabling designers to quickly see

the effects of any changes to the design architecture and informing design decisions as part of an iterative process.

However, even with this automation, manual evolution of the design is still a complex and time-consuming process, relying on experimentation with different design options on the basis of the designers' experience and intuition, and may still be difficult to fulfil the competing requirements on the system — particularly as the time and effort involved means that typically only a small number of potential design options can be investigated in this way.

One potential solution to this type of problem is automatic architectural optimisation. This allows a much more efficient search of the design space (the total set of possible design variations), covering a much greater number of alternatives than could be considered manually. However, to be effective, the optimisation must be guided in some way, which requires some form of heuristic evaluation. The use of ADLs such as EAST-ADL pays dividends at this point, because such a model contains all of the information necessary for analysis to take place with respect to each system attribute being considered. For example, if the objectives of the optimisation were to maximise safety and performance while minimising cost and unavailability, it has to be possible to evaluate each of the candidate designs being explored according to those four characteristics by means of some form of analysis. By making changes to the design model automatically as part of the optimisation process and allowing the analysis tools to access the same common model, it then becomes possible to evaluate the new variant directly and seamlessly, facilitating a rapid, efficient search of the design space.

3.1 Multi-Objective Optimization

When more than one objective is being optimised, e.g. both performance and cost, it becomes a *multi-objective optimisation* problem. Optimising a large potential design space to obtain one or more good solutions that feature a desirable balance of attributes is generally more difficult, as the objectives may conflict such that improving one objective worsens another (e.g. higher performance or better safety may mean greater cost). There are a variety of different automated algorithms that can be employed to solve such problems, but in general they all aim to quickly find viable solutions (i.e., valid design variants in this case) that offer optimal or near-optimal attributes without needing to investigate all possible designs. Most of these techniques do not typically produce a single optimum solution; instead, the goal is often to produce a set of 'optimal' solutions that feature a range of attributes that balance the different objectives in different ways — in other words, each providing a different trade-off between the objectives whilst still meeting any constraints.

These are known as the *Pareto solutions* and are based on the concept of *dominance*. A dominant solution is one that is better in at least one objective than every other solution yet found, and no worse in the other objectives. The set of dominant solutions is known as the Pareto set. When plotted on the graph, they tend to form a curve known as the Pareto frontier, which shows the current progress of the optimisation; new solutions beyond the Pareto frontier are new optimal solutions and will

dominate older solutions, whereas solutions within the frontier are dominated and non-optimal. This is shown in Figure 2, where the Pareto frontier consists of the solutions marked as black dots, whilst dominated solutions are white dots.

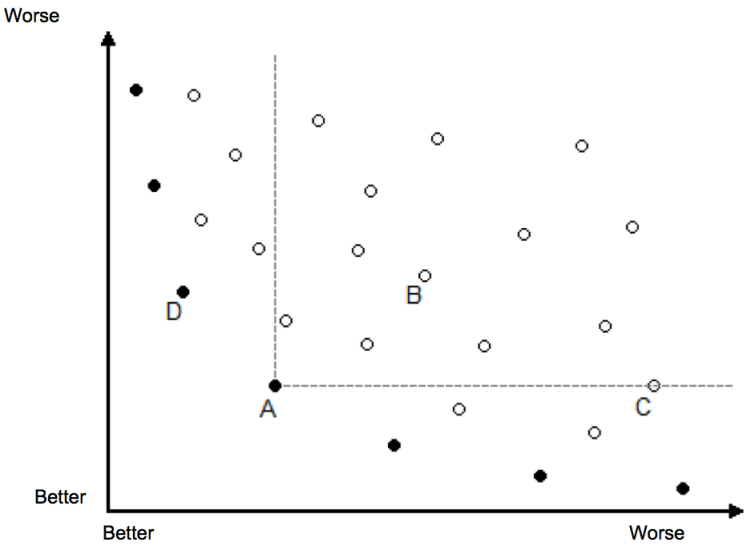

Figure 2. Pareto frontier.

In Figure 2, solutions are shown as dots plotted against two axes, each representing different objective attributes (e.g. unavailability and cost). In this case, the goal is to minimise each attribute, so lower values are better. Thus A dominates B because it has lower values for each attribute; it is better in both respects. A also dominates C, because although they have the same value for one objective (e.g. unavailability), A has a lower value for the other (e.g. it is cheaper). Conversely D is a non-dominated solution because it is better than A in one objective but worse in another; for example, D may be cheap but not very reliable, whereas A is more reliable but also more expensive — both are equally valid solutions.

Many different multi-objective optimisation algorithms exist, e.g. tabu search [KS1], ant colonies [LS1], and simulated annealing [KB1]. In the MAENAD project, we have chosen another prominent approach based on *genetic algorithms*. Genetic algorithms are optimisation algorithms inspired by the evolutionary processes found in nature based on the idea of starting with one or more randomly chosen candidates and then evolving the next candidates for further evaluation from previous candidates through mutation. Genetic algorithms have been identified as a particularly effective method for solving combinatorial optimisation problems (such as those discussed here) and they are capable of navigating large, complex search spaces [CS1]. In particular, a variation of Non-Dominated Sorting Genetic Algorithm 2, NSGA-II [DP1] has been applied in the MAENAD project.

3.2 EAST-ADL Optimization Architecture

The EAST-ADL optimisation tool architecture is currently being developed as part of the MAENAD project; more information can be found in our more detailed account [WR1], but a summary of our approach is provided here.

The tool architecture is designed to leverage the information-centralising capabilities of EAST-ADL to integrate with and provide the relevant information to the various analysis and optimisation tools required. A prototype optimisation environment has been developed based on the EPM tool [AL1], realized as a plugin for the Eclipse framework, which currently implements a core subset of EAST-ADL.

The three main aspects of the tool architecture are as follows:

Design space definition: In order for optimisation to take place, it must be possible to define the design space by means of specifying variation points in the model that the optimisation can later choose and evaluate. This is primarily achieved through the use of EAST-ADL's variability management mechanism, which allows alternative implementations and allocations to be specified. In our prototypical implementation of the optimization architecture, this role is fulfilled by the EPM tool.

Objective evaluation: For optimisation to be guided rather than random, it must be possible to evaluate new solutions to determine whether they are better or worse than previous solutions. In a multi-objective problem, this means analysing the solutions (in the form of design candidates) for each of the objectives being evaluated. In this case, HiP-HOPS is the primary tool to evaluate safety and reliability characteristics, cost is handled by a dedicated Eclipse plugin (currently only a simplistic summation), and timing analysis capabilities are provided by the MAST tool (Modelling and Analysis Suite for Real-Time Applications, http://mast.unican.es/). Other analyses can be added as needed to evaluate other objectives.

Efficient design space exploration: Finally, optimisation requires that the design space be explored by means of an algorithm. In this case, a variant on the NSGA-II genetic algorithm is used, based on prototype optimisation technology in HiP-HOPS but implemented as an EPM plugin instead. In combination with the design space definition and analysis capabilities, it allows automatic optimisation of an EAST ADL architecture.

The final piece of the puzzle is *variability resolution*. Before a design variant (as selected by the optimisation algorithm) can be analysed and evaluated, the variability first has to be 'resolved', i.e., to produce a model in which all of the variability points have been selected or deselected to produce a particular system configuration with a concrete set of objective attributes.

Even with the advantages presented by using an EAST-ADL model as the core of the process, there are still certain issues and constraints to be dealt with. For example, EAST-ADL does not require a one-to-one mapping between its nominal and error model architectures, i.e., it does not ordinarily require that each function has a corresponding error model type; however, for the error model to be evaluated correctly when the nominal architecture changes, the optimisation requires a one-to-one mapping to be applied. This is resolved by explicitly defining the Error Model for each nominal model variant. Furthermore, the optimisation process cannot work unless the

optimisation variability guarantees substitutability — that selection of one option over another does not invalidate the system. If this was not the case, then it would mean that the optimisation algorithm could produce nonsensical or impossible designs, e.g. components that are no longer able to be connected together because they do not have compatible interfaces. Substitutability therefore ensures that the resulting system design remains valid and sensible, and consequently this imposes two requirements on the variants: that they share a compatible interface (i.e., they share at least a common subset of ports and connections) and that they perform equivalent functions. This is not necessarily simple to achieve, especially in an ADL such as EAST-ADL, where the error model, software, and hardware may all be represented in separate model views and a change in one may need to be reflected in the others. At the same time, this is a criterion for validity of the model regardless of context: it is not only optimisation which requires that a valid timing model (or error model or any other model) must emerge together with the core architecture as a result of changes to the variability; this is a general issue in any variability resolution process.

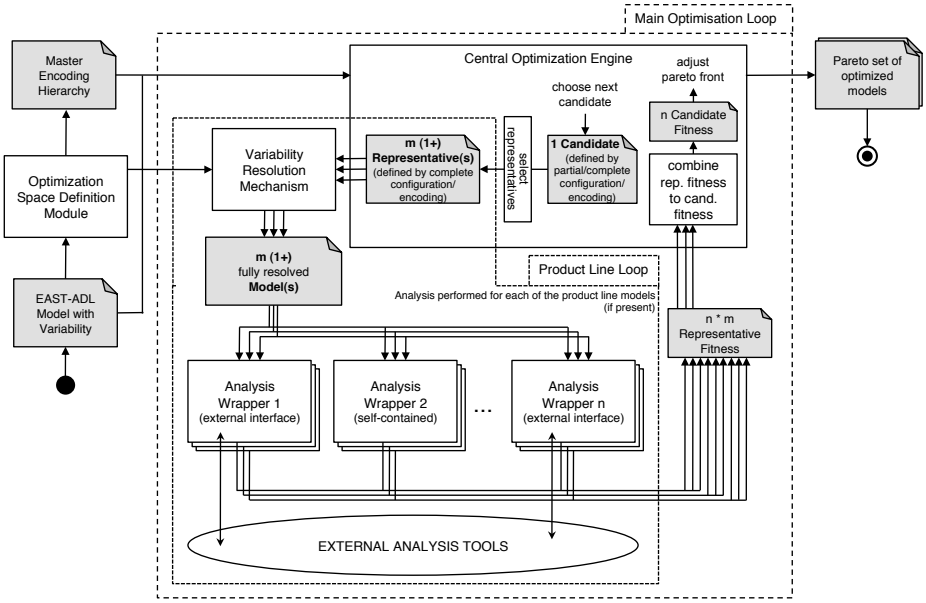

Figure 3. EAST-ADL Optimisation Architecture.

Figure 3 below illustrates the current tool architecture as defined in the MAENAD project. This diagram shows the flow of operations in the optimisation process (white boxes) and the main entities of information passed between these operations (grey boxes). The initial EAST-ADL model, variability that defines the design space, is passed to the Optimisation Space Definition Module (OSDM), which generates a Master Encoding Hierarchy. This defines the search space for the Central Optimisation Engine (COE), which generates different optimisation candidates in the form of

model encodings / model configurations that are passed to the Variability Resolution Mechanism (VRM) to be resolved into analysable models. The analysis wrappers then evaluate these models according to different objectives and return the results to the COE. Eventually the optimisation settles on a set of Pareto optimal solutions, which constitute the final results of the process.

3.3 Product Line Optimization

Figure 3 also includes elements to support *product line variability*. Variability in EAST-ADL was originally included to help cater for different product lines that may have different sets of features but share an overall commonality of architecture. Over the past months, the optimisation process has been extended to work with not just a single member of a product line, but several or all members of a product. Because product lines are already represented by the same sort of EAST-ADL variability management techniques used to define the design space for optimisation, it was possible to include both optimisation and product line variability in the same model (with the two forms of variability distinguished by the binding time for each variation), and exhaustively or selectively analyse the variants of the product line as part of the overall optimisation loop, shown as "Product Line Loop" in Figure 3.

The major complication is that analysing an entire product line is different to analysing a single product; present analysis tools only analyse one product at a time, meaning that the analysis plugins have to iterate over the product line and combine the results in some form (cf. box "combine rep. fitness to cand. fitness" in Figure 3). Whilst in some cases this only involves a simple summation of the individual fitness values or the computation of an arithmetic mean, in other cases the combination may involve a more complex calculation that would have to be carried out by a non-trivial component of the optimisation architecture. For example, cost may be determined by the sum of each piece cost multiplied by the piece count for that piece (where the number of pieces is potentially different for each product as well as each design candidate), whereas for something like safety, the combinatory unit may simply use a maximum unavailability value as its result instead. This has to be determined on a case-by-case basis, as each analysis has different. Currently, the prototypical implementation developed in the MAENAD project supports only a combination function that is hardcoded in Java. While this provides the maximum flexibility for our current experimentation, it is, of course, required to allow changes to the combination function without changes to the code at a later point in time. In general, product line optimisation is still being investigated further in the MAENAD project and not all details have been fixed yet.

4 Related Work

There is a range of other work being conducted on architecture based optimisation. One branch of this work has focused on safety and reliability analysis, though not in the context of ADLs; for example, classical dependability models like Reliability

Block Diagrams (RBDs) [KC1] and, more recently, advanced compositional dependability analysis techniques such as HiP-HOPS have been combined with meta-heuristics (Pareto-based Genetic Algorithms) to assist in the automatic evolution of design models that can meet dependability and cost requirements. HiP-HOPS has contributed to this area by enabling the optimisation of systems that have a networked architecture (i.e. they are not necessarily in parallel/series configurations as in RBDs) and by overcoming the traditional assumption made in RBDs that a component or system either works or fails in a single failure mode [PW1,AP1].

Other work has focused on development of tools for multi-objective optimisation of software architectures. One such tool is PerOpteryx, which is based on the Palladio modelling environment [MA1,KR1]. PerOpteryx allows for the automatic optimisation of software architecture models, developed with Palladio using the Palladio Component Model (PCM), on the basis of four main quality dimensions: cost, reliability, maintainability, and performance. One particular advantage of PerOpteryx is its ability to take advantage of domain specific knowledge, such as performance tactics, to enhance the optimisation [KK1].

Another tool is AQOSA (Automated Quality-driven Optimisation of Software Architecture), which uses model transformation technology to convert input models (e.g. from AADL or a general UML2 model) into an intermediate format (AQOSA-IR) that can be used as the basis of the optimisation process [EC1]. Different candidates are provided by a repository of possible components, and a set of external objective function plugins provides the evaluation that drives the process. AQOSA is designed to be independent of any given domain specific language (DSL) or ADL, but therefore relies on a correct model transformation to its own intermediate model to perform the optimisation.

In addition, research by Grunske et al. has shown the potential of using various meta-heuristics for dependability versus cost optimisation of architectural designs. This work has been applied to AADL models and also looks at other optimisation approaches beyond genetic algorithms, e.g. ant trail algorithms [MM1]. An introduction to the model-based optimisation field can also be found in [GL1], and a wider survey of literature on architectural optimisation techniques can be found in [AB1].

While the above approaches all exploit meta-heuristics to search a design space for optimal solutions, like our approach, they all operate on different input models (EAST-ADL, PCM, AQOSA-IR etc.) and use different means of defining the design space (i.e. different sorts of variability mechanisms). They also address different objectives, which they define in different ways, and use different techniques to evaluate those objectives. Consequently, each approach has different strengths and weaknesses depending on the domain and the optimisation objectives in question.

The optimisation work conducted as part of the MAENAD project aims to contribute to the state of the art by developing an approach which makes use of a combination of analysis techniques (e.g. HiP-HOPS, MAST) derived from information obtained from the variability capabilities and quality attributes provided by EAST-ADL. Therefore, while there are both commonalities and differences across the optimisation approaches mentioned above, we believe the approach we are working on offers a

unique configuration that combines the benefits of external analysis tools and the all-in-one modelling capabilities offered by EAST-ADL.

It should be noted that this paper is intended to serve as a brief summary of a more detailed article on the MAENAD optimisation approach [WR1], but it also includes an account of the latest improvements and additions developed in the project, particularly in terms of the product line optimisation work.

5 Outlook

The MAENAD project will be completed in February 2014. Until then we are planning to refine and finalize the optimisation architecture based on lessons learned from the several modelling experiments and case studies in the MAENAD project. One main focus of attention during the final months of the project will be the support for product line variability in the context of multi-objective optimisation. It is believed that this will offer a unique capability that will make use of the domain-specific information provided by EAST-ADL to support the design not just of single products, but entire product lines as well.

Acknowledgements. This work has been supported financially by the European Commission under grant agreement number 260057.

References

[AB1] Aleti A., Buhnova B., Grunske L., Koziolek A., Meedeniya I. (2012) "Software Architecture Optimization Methods: A Systematic Literature Review. In *IEEE Transactions on Software Engineering*, Issue 99, September 2012. ISSN: 0098-5589, DOI: 10.1109/TSE.2012.64

[AL1] Abele A., Lönn H., Reiser M-O., Weber M., Glathe H. (2012) EPM: a prototype tool for variability management in component hierarchies. In *Proceedings of the 16th International Software Product Line Conference - Volume 2 (SPLC'12)*. Published by the ACM, New York, USA, 2012, pp 246-249. ISBN: 978-1-4503-1095-6, DOI: 10.1145/2364412.2364455

[AP1] Adachi M., Papadopoulos Y., Sharvia S., Parker D., Tohdo T. (2011) An approach to optimization of fault tolerant architectures using HiP-HOPS, Software Practice and Experience, DOI: 10.1002/spe.104436 pages, Wiley Interscience

[CS1] Coit, D. W. and Smith A.E. (1996a). Penalty guided genetic search for reliability design optimisation. *Computers and Industrial Engineering*. 30(4), pp.895-904.

[DP1] Deb, K, Pratap A, Agarwal S, and Meyarivan T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*. 6(2), pp.182-197.

[EC1] Etemaadi R., Chaudron M.R.V. (2012) A model-based tool for automated quality-driven design of system architectures. In *Proceedings of the 8th*

European Conference on Modelling Foundations and Applications (ECMFA'12), 2-5 July 2012, Lyngby, Denmark.

[GL1] Grunske L., Lindsay P., Bondarev E., Papadopoulos Y., Parker D., (2007) "An Outline of an Architecture-Based Method for Optimizing Dependability Attributes of Software-Intensive Systems", R. de Lemos et al. (Eds.): Architecting Dependable Systems IV, LNCS 4615, 2007, pp. 188-209

[KB1] Kim, H, Bae C., and Park S. 2004. Simulated annealing algorithm for redundancy optimization with multiple component choices. In: Advanced Reliability Modelling, Proceedings of the 2004 Asian Internationals Workshop. World Scientific, pp. 237-244.

[KC1] Konak, A., Coit, D.W., and Smith, A.E., (2006) Multi-objective optimization using genetic algorithms, *Reliability Engineering & System Safety*, 91(9):992-1007, Elsevier

[KK1] Koziolek A., Koziolek H., Reussner R. (2011). PerOpteryx: automated application of tactics in multi-objective software architecture optimisation. In *Proceedings of the joint ACM SIGSOFT conference -- QoSA and ACM SIGSOFT symposium -- ISARCS on Quality of software architectures -- QoSA and architecting critical systems -- ISARCS*, New York, USA, 2011. pp. 33-42. ISBN: 978-1-4503-0724-6, DOI: 10.1145/2000259.2000267

[KR1] Koziolek A. and Reussner R. (2011). Towards a generic quality optimisation framework for component-based system models. In *Proceedings of the 14th International ACM SIGSOFT Symposium on Component Based Software Engineering (CBSE'11)*, 21-23 June 2011, Boulder, Colorado, USA.

[KS1] Kulturel-Konak S., Smith A.E, Coit D.W. (2003). Efficiently solving the redundancy allocation problem using tabu search. *IIE Transactions*. 35, pp. 515-526.

[LS1] Liang, Y C and Smith A E. (2004). An ant colony optimisation algorithm for the redundancy allocation problem (RAP). *IEEE Transactions on Reliability*. 53(3), pp. 417-423.

[MA1] Martens A., Adagna D., Koziolek H., Mirandola R., Reussner R. (2010) A hybrid approach for multi-attribute QoS optimisation in component-based systems. *Proceedings of the 6th International Conference on the Quality of Software Architectures (QoSA'10)*, Lecture Notes in Computer Science Vol 6309, pp 84-101. Springer Berlin Heidelberg. ISBN: 978-3-642-13821-8, DOI: 10.1007/978-3-642-13821-8_8

[MM1] Meedeniya I., Moser I., Aleti A. and Grunske L. (2011) Architecture-Based Reliability Evaluation under Uncertainty. In Quality of Software Architectures (QoSA 2011), Boulder, Colorado, USA, June, 2011, pp. 85-94.

[PW1] Papadopoulos Y., Walker M., Parker D., Rüde E., Hamann R., Uhlig A., Grätz U., Lien R. (2011) Engineering Failure Analysis & Design Optimisation with HiP-HOPS, Journal of Engineering Failure Analysis, DOI: 10.1016/j.engfailanal.2010.09.025, Elsevier Science, ISSN: 1350 6307.

[WR1] Walker M., Reiser M-O., Tucci-Piergiovanni S., Papadopoulos Y., Lönn H., Mraidha Ch., Parker D., Chen D-J., Servat D. (2013) "Automatic optimisation of system architectures using EAST-ADL". In *Journal of Systems and Software*, to appear (available online). DOI: 10.1016/j.jss.2013.04.001